REMARKS

Claims 1, 2, 5-7, 9, 12-15, and 18-25 are pending in the present Application. Applicant appreciates Examiner's explanations in the Response to Arguments.

Examiner has rejected claims 1, 2, 9, 12, 13, 15, 18, 23, and 25 under 35 U.S.C.§103(a) as being unpatentable over U.S. Patent No. 6,470,329 to Livschitz ("Livschitz") in view of U.S. Patent Application Publication No. 2002/0029214 by Yianilos et al. ("Yianilos").

Livschitz is directed to the synchronization of a data set and a remote copy of the data set. Livschitz teaches that physical data are fixed-length records in data files that can be decomposed into elementary data blocks that permit no further decomposition. See col. 5, lines 55-60. Livschitz teaches that a hash function may operate upon the data sets and the subsets of data sets, and that this hash function is available in two address spaces (e.g., local and remote copies of a data set). See col. 5, lines 63-65 and Figs. 1 and 8. Livschitz describes at least three processes of synchronization. First, a hash function, H, operates upon separated data sets, A and B, to create signatures (e.g., hashes) h(A) and h(B). These hash signatures are compared and if there is a mismatch, the data sets are each subdivided and again hashed with H followed by the subdivided data set hashes being compared. This process is repeated recursively until the portions of all of the data sets needing synchronization are identified. See col. 5, line 66 - col. 6, line 12. Livschitz discloses a second synchronization process in which the hash signature for each elemental data block is created and stored in an array of h(A) and h(B) signatures, respectively. Each hash signature h(A) is communicated and compared to hash signature h(B) to determine which elemental block requires synchronization. See col. 7, lines 31-48. Livschitz

acknowledges that this process may consume a large amount of bandwidth. See col. 7, lines 57-59. A third process is disclosed in which the hash signature of each elemental data block (h(A), h(B)) is precalculated and stored in a respective array. Each array is then subjected to a second hash function, G, to generate a hash signature (g(h(A)), g(h(B))) of the array of hash signatures. The g(h(A)) signature is communicated and compared to g(h(B)) to isolate the elements of h(A) and h(B) that are different. See col. 7, line 64 - col. 8, line 19.

Examiner has stated that when an out of match condition is determined ("If the signature g(hA) is not identical to g(hB)..."), a second hash is generated. Applicants respectfully submit that this is not an accurate portrayal of the teaching of Livschitz. Livschitz generates a second hash using function G and the G hash is calculated from the first hash signature from function H. However, such a Livschitz second hash is not conditional upon the H function hash signatures being compared and an out of match condition recognized therefrom. Rather, Livschitz teaches that the G function hash proceeds directly from the H function hash signatures being stored in an array and the comparison is only of G function hashes.

Moreover, Livschitz teaches that the first hash is an operation on the data set, itself, (or a subset thereof) and that the second hash is an operation on the first hash. To the contrary, Applicants have claimed that Applicants' first hash is based upon database *values* and Applicants' second hash is based upon database *records*. Also, Applicants have claimed that the "second computational intensity is greater than said first computational intensity and requires a greater amount of communication channel capacity to communicate said second hash than said first

hash". Livschitz does not disclose this feature, either, and in fact acknowledges that the recursive hash communication may consume a large amount of bandwidth.

Examiner has acknowledged that Livschitz does not disclose generating "a first hash pursuant to a first hash technique of a first computational intensity and based on the database values...and communicat[ing] said first hash to the network part" and generating "a second hash pursuant to a second hash technique of a second computational intensity... and communicat[ing] said second hash to the network part... in which said second computational intensity is greater than said first computational intensity and reqiures a greater amount of communication channel capacity to communicate said second hash than first hash." Yianilos has been introduced as teaching this missing element.

Yianilos is directed to a database system having the ability to synchronize data within a specified key range between a primary host database and a secondary host database via a limited bandwidth link. See paragraphs [0002], [0005], and [0080]. Applicant agrees with Examiner that Yianilos teaches two hash techniques, a "Get_All_Hashes" function and a "Get_Interval_Hashes" function. The "Get_All_Hashes" function produces pairs in the two-part form of the key field of the record + a fixed size digest (hash) of the record over the records in interval I to K. See paragraph [0070]. The "Get_Interval_Hashes" function produces a list of "triplets" in the form of the sub-interval (key_interval) + the number of records in the sub-interval (num_records) + a fixed size digest of all of the records in the database (hash) over a defined interval of I,K having H disjoint subintervals. Examiner interprets Yianilos as teaching the Get_All_Hashes as being more computationally intensive than the Get_Interval_Hashes.

Yianilos teaches a specific implementation of the two hashing functions at paragraphs [0080]-[0083]. Yianilos employs a limited bandwidth link between separated databases and attempts to limit the number of bits transferred. When synchronizing the databases, both databases are caused to use the Get_Interval_Hashes function to compute a single summary of all of the records lying in a key interval. Since a decision is to be made at the local side whether or not a match is present over the key interval of record summary, each of the outputs from the Get_Interval_Hashes functions over each of the key intervals at the remote database must be conveyed from the remote database to the local database for comparison (that is, a conveyance for each H sub-interval). If a mismatch is discovered, "the size of the remote database restricted to the given key interval is checked. If the remote database only has a small number of records lying in the key interval, then digests for all those individual records are transferred from the remote to the local side (the Get_All_Hashes function is invoked here), and a record-by-record comparison is made to identify discrepancies." Paragraph [0083]. If the size of the remote database in the key interval has a large number of records "the remote database is asked (by calling the Get_Interval_Hashes function) to partition the key range into smaller sub-intervals and send summaries for each of the sub-intervals. These remote summaries are then compared against corresponding local summaries and the operation is invoked recursively for sub-interval whose summaries do not match." Paragraph [0083]. In the case of a large number of records, the same hashing technique is reused and, as Applicant has previously pointed out, there is no difference in computational complexity. Of more interest relative to Applicant's claims is the case where the number of records is small and the Get_All_Hashes function is invoked. In the instance where a plurality of records are involved, the invocation of the Get_All_Hashes function

may have a greater computational intensity and require a greater amount of communication channel capacity than the first use of the Get_Interval_Hashes function. Yianilos discloses that the invoking of the Get_All_Hashes function must be carefully done, since choosing a key interval that includes a larger number of records reduces the number of times a Get_All_Hashes output must be produced and conveyed over Yianilos's limited bandwidth. However, each such round generates heavy traffic on Yianilos's limited bandwidth link. Choosing a key interval that includes fewer records increases the number of communication rounds but with a lighter traffic load for each round. See paragraph [0085]. Briefly stated, for certain circumstances Yianilos teaches the sending of a *plurality* of hashed outputs in conveying a first hash of a first computational complexity and the sending of a *plurality* of hashed outputs in conveying a second hash of a second computational complexity. Applicant's claimed invention has been clarified to indicate each of Applicant's first hash and the second hash are sent in a respective first message and a second message, unlike Yianilos and consistent with Applicant's disclosure at paragraphs [0047] and [0048] and Fig. 3.

All of the elements of Applicants' claim 1, therefore, have not been disclosed by Livschitz or Yianilos, taken alone or in combination. For this reason, a *prima facie* case of obviousness has not been made and Applicants believe claim 1 to be allowable over the cited art. For the same reasons, independent claims 15 and 23 are also believed allowable. Dependent claims 2, 9, 12, 13, 18, and 25 are dependent upon presumed allowable independent claims and are themselves presumed allowable for this reason.

Dependent claims 21, 22, and 24 have been rejected under 35 U.S.C.§103(a) as being unpatentable over Livschitz and Yianilos and further in view of U.S. Patent Application Publication No. 2002/0120648 by Ball ("Ball"). Examiner has rejected claims 5-7 under 35 U.S.C.§103(a) as being unpatentable over Livschitz and Yianilos and further in view of U.S. Patent No. 5,809,494 to Nguyen. Examiner has also rejected claims 14, 19, and 20 under 35 U.S.C.§103(a) as being unpatentable over Livschitz and Yianilos and further in view of U.S. Patent No. 5,684,990 to Boothby. Claims 5-7, 14, 19, 20-22, and 24 are dependent upon presumed allowable independent claims and, by virtue of such dependency, are themselves presumed allowable.

In light of the foregoing amendment and remarks, Applicant believes all of the pending claims to be allowable. Examiner is respectfully requested to enter the present Amendment, withdraw the claims rejection, reconsider the present Application, and pass the present Application to allowance. In the alternative, Examiner is respectfully requested to enter the present Amendment as placing the present Application in a condition better suited for appeal.

Respectfully submitted,

/ Robert H. Kelly /

Robert H. Kelly
Registration No. 33,922

KELLY & KRAUSE, L. P.
6600 LBJ Freeway, Suite 275
Dallas, Texas 75240
Telephone: (214) 446-6684
Fax: (214) 446-6692